WHAT IS CLAIMED IS:

1. A method for using runtime drivers during pre-boot, comprising:

booting a computing system;

initializing memory in the computing system;

initializing at least one pre-boot mapping driver;

for each device requiring pre-boot operation, identifying whether the device's driver is a runtime driver needing to be mapped to a pre-boot driver; and

for each identified runtime driver,

binding the identified runtime driver to a pre-boot mapping driver to generate a mapped runtime driver image;

loading the mapped runtime driver image; and

starting the mapped runtime driver image.

2. The method as recited in claim 1, wherein the pre-boot mapping drivers are compatible with an extensible firmware interface (EFI).

3. The method as recited in claim 1, wherein the runtime drivers are selected from a group consisting of Windows™ drivers, Linux drivers, fcode drivers, and EFI drivers.

4. The method as recited in claim 1, wherein identifying whether the driver is a runtime driver needing to be mapped to a pre-boot driver, comprises:

accessing a header section of a runtime image; and determining an image type and subsystem type associated with the runtime image, wherein if the subsystem type is EFI then mapping is not performed.

5. The method as recited in claim 1, further comprising booting an operating system (OS) loader.

6. The method as recited in claim 1, wherein the pre-boot driver is a firmware extension.

7. The method as recited in claim 1, wherein binding the identified runtime driver comprises binary rewriting of system calls.

8. The method as recited in claim 1, wherein binding the identified runtime driver comprises:

intercepting system calls; and

mapping the system calls to service calls compatible with the pre-boot infrastructure.

9. The method as recited in claim 8, wherein the pre-boot infrastructure is an extensible firmware interface (EFI).

10. The method as recited in claim 1, wherein a runtime driver to be used as a pre-boot driver is selected based on size and efficiency of the runtime driver.

11. An article of manufacture comprising a machine accessible medium containing code having instructions that, when executed during pre-boot, cause the machine to:

initialize memory in the computing system;

initialize at least one pre-boot mapping driver;

for each device requiring pre-boot operation, identify whether the device's driver is a runtime driver needing to be mapped to a pre-boot driver; and

for each identified runtime driver,

bind the identified runtime driver to a pre-boot mapping driver to generate a mapped runtime driver image;

load the mapped runtime driver image; and

start the mapped runtime driver image.

12. The article as recited in claim 11, wherein the pre-boot mapping drivers are compatible with an extensible firmware interface (EFI).

13. The article as recited in claim 11, wherein the runtime drivers are selected from a group consisting of Windows™ drivers, Linux drivers, fcode drivers, and EFI drivers.

14. The article as recited in claim 11, wherein identifying whether the driver is a runtime driver needing to be mapped to a pre-boot driver, comprises:

accessing a header section of a runtime image; and

determining an image type and subsystem type associated with the runtime image, wherein if the subsystem type is EFI then mapping is not necessary.

15. The article as recited in claim 11, wherein the code further comprises instructions that boot an operating system (OS) loader.

16. The article as recited in claim 11, wherein the pre-boot driver is a firmware extension.

17. The article as recited in claim 11, wherein binding the identified runtime driver comprises binary rewriting of system calls.

18. The article as recited in claim 11, wherein binding the identified runtime driver comprises instructions that:

intercept system calls; and

map the system calls to service calls compatible with the pre-boot infrastructure.

19. The article as recited in claim 18, wherein the pre-boot infrastructure is an extensible firmware interface (EFI).

20. The article as recited in claim 11, wherein a runtime driver to be used as a pre-boot driver is selected based on size and efficiency of the runtime driver.

21. A system comprising:

platform hardware comprising a processor coupled with system memory and pre-boot memory;

an extensible firmware interface (EFI) core infrastructure to enable communication among the processor and a plurality of hardware devices coupled to the platform hardware;

at least one hardware device driver, wherein the at least one hardware device driver is required during pre-boot; and

an EFI driver wrapper to enable the at least one hardware device driver to operate during pre-boot.

22. The system as recited in claim 21, wherein a hardware device driver designed for a runtime environment is associated with the EFI driver wrapper using binary rewriting.

23. The system as recited in claim 21, wherein a hardware device driver designed for a runtime environment is associated with the EFI driver wrapper using system call remapping.

24. A method for mapping a driver for use in an alternative operational environment, comprising:

booting a computing system;

initializing memory in the computing system;

initializing at least one alternative operational environment mapping driver;

for each device requiring alternative operational environment operation, identifying whether the device's driver is a driver needing to be mapped to a alternative operational environment driver; and

for each identified driver needing mapping,

binding the identified driver to an alternative operational environment mapping driver to generate a mapped driver image;

loading the mapped driver image; and

starting the mapped driver image.

25. The method as recited in claim 24, wherein the mapping drivers are compatible with an extensible firmware interface (EFI).

26. The method as recited in claim 25, wherein the drivers are selected from a group consisting of Windows™ drivers, Linux drivers, fcode drivers, and EFI drivers.

27. The method as recited in claim 25, wherein identifying whether the driver is a driver needing to be mapped to am alternative operational environment driver, comprises:

accessing a header section of a driver image; and

determining an image type and subsystem type associated with the driver image, wherein if the subsystem type is EFI then mapping is not performed.

28. The method as recited in claim 24, further comprising booting an operating system (OS) loader.

29. A system comprising:

platform hardware comprising a processor coupled with system memory and pre-boot memory;

a core infrastructure to enable communication among the processor and a plurality of hardware devices coupled to the platform hardware;

at least one hardware device driver, wherein the at least one hardware device driver is required during a first operational execution environment; and

a driver wrapper to enable the at least one hardware device driver to operate during an alternative operational execution environment.

30. The system as recited in claim 29, wherein a hardware device driver designed for the first operational execution environment is associated with the driver wrapper using binary rewriting.

31. The system as recited in claim 29, wherein a hardware device driver designed for the first operational execution environment is associated with the driver wrapper using system call remapping.